**make xxx\_defconfig**

Câu lệnh cấu hình cho board Beagle Bone Black là :

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf am335x\_boneblack\_vboot\_defconfig

Chúng ta sẽ cùng đi vào phân tích câu lệnh này để tìm ra được cách hoạt động của quá trình ghi xxx\_defconfig

**Phiên Bản U-boot**

1. VERSION = 2019
2. PATCHLEVEL = 07
3. SUBLEVEL =
4. EXTRAVERSION = -rc4
5. NAME =

Thông tin về phiên bản được đặt ngay phần đầu của top Makefile.

Số hiệu phiên bản của Uboot được chia làm 3 cấp :

VERSION: Số hiệu phiên bản chính.

PATCHLEVEL: Số hiệu phiên bản phụ.

SUBLEVEL:

EXTRAVERSION: Thông tin phiên bản bổ sung.

Ví dụ : uboot\_5.2.1

**Thiết Lập Môi Trường Cho Máy Chủ Lưu Trữ**

Vì được xây dừng riêng cho việc phát triển các hệ thống nhúng, do đó một trong các việc cần phải làm đó chính là xác định môi trường xây dựng của máy chủ bao gồm : Lấy ra ARCH của OS và SHELL.

Đoạn mã dưới đây nhằm mục đích lấy ra thông tin kiến trúc của máy tính hiện tại.

HOSTARCH := $(shell uname -m | \

sed -e s/i.86/x86/ \

-e s/sun4u/sparc64/ \

-e s/arm.\*/arm/ \

-e s/sa110/arm/ \

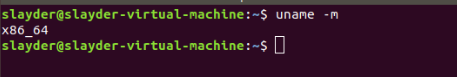
-e s/ppc64/powerpc/ \

-e s/ppc/powerpc/ \

-e s/macppc/powerpc/\

-e s/sh.\*/sh/)

$(shell uname –m) có nghĩa là thực thi uname –m bằng shell scripts và lấy ra kiến trúc của máy chủ, thực thi trên máy ảo ubuntu và kết quả trả về là x86\_64.



sed –e s/xxx/yyy có nghĩa là tìm xem có kết quả xxx nào không, nếu có hãy thay thế nó bằng yyy.

Ví dụ : sed –e s/i.86/x86 sẽ tìm kiếm có chuỗi i.86 trong đó hay không. Nếu có thì sẽ thay thế bằng x86

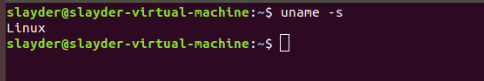
Ngoài ra thì .\* là kí tự đại diện. Kết quả của uname –r ở máy tính này là x86\_64 nên nó không bị thay thế bằng các ký tự khác. Do đó

HOSTARCH:= x86\_64

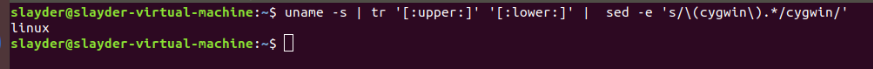
HOSTOS := $(shell uname -s | tr '[:upper:]' '[:lower:]' | \

sed -e 's/\(cygwin\).\*/cygwin/')

$(shell uname –s) là thực thi uname –s bằng shell scripts và lấy ra hệ điều hành của máy chủ, thực thi trên máy ảo ubuntu và kết quả trả về là Linux.



Sau khi lấy được tên máy chủ, tiếp theo sẽ thay thế hết tất cả các chữ in hoa bằng các chữ thường và kết quả thu được là linux.



export HOSTARCH HOSTOS

Câu lệnh export tương đương với extern trong ngôn ngữ C, biến các biến HOSTARCH và HOSTOS thành các biến môi trường, giúp cho bạn có thể sự dụng ở bất kì đâu trong hệ thống.

**Chỉ Định Thư Mục Lưu Trữ**

Hệ thống Kbuild cho phép lưu trữ file ở một thư mục riêng. Để xác định vị trí các tệp đầu ra trong một thư mục riêng biệt, có hai cú pháp được hỗ trợ :

1. O=

Use "make O=dir/to/store/output/files/"

1. Set KBUILD\_OUTPUT

Thiết lập biến môi trường KBUILD\_OUTPUT trỏ tới thư mục muốn lưu trữ các tập tin sẽ được cài đặt.

export KBUILD\_OUTPUT=dir/to/store/output/files/

Việc sử dụng O= được ưu tiên hơn so với việc sử dụng biến môi trường KBUILD\_OUTPUT.

Nếu hai cách trên không được sử dụng thì thư mục hiện tại sẽ là nơi lưu trữ.

ifeq ("$(origin O)", "command line")

KBUILD\_OUTPUT := $(O)

endif

$(origin O) Kiểm tra xem biến O có xuất phát từ command line hay không. Nếu O được gán thì KBUILD\_OUTPUT := $(O)

ifneq ($(KBUILD\_OUTPUT),)

# Invoke a second make in the output directory, passing relevant variables

# check that the output directory actually exists

saved-output := $(KBUILD\_OUTPUT)

KBUILD\_OUTPUT := $(shell mkdir -p $(KBUILD\_OUTPUT) && cd $(KBUILD\_OUTPUT) \

&& /bin/pwd)

$(if $(KBUILD\_OUTPUT),, \

$(error failed to create output directory "$(saved-output)"))

ifneq ($(KBUILD\_OUTPUT),) nhằm mục đích kiểm tra xem cả 2 phương pháp O= và KBUILD\_OUTPUT có được sử dụng hay là không. Nếu KBUILD\_OUTPUT không rỗng thì saved-output được lưu bằng giá trị $(KBUILD\_OUTPUT).

Nếu không thì sẽ tự tạo thư mục đầu ra.

KBUILD\_OUTPUT := $(shell mkdir -p $(KBUILD\_OUTPUT) && cd $(KBUILD\_OUTPUT) \

&& /bin/pwd)

Tạo KBUILD\_OUTPUT nếu nó không tồn tại. Tham số -p của mkdir thể hiện nếu KBUILD\_OUTPUT không tồn tại (rỗng). cd $(KBUILD\_OUTPUT) là thử nhập xem có được không. Nếu được gán đường dẫn tệp cho KBUILD\_OUTPUT.

$(if $(KBUILD\_OUTPUT),, \

$(error failed to create output directory "$(saved-output)"))

Thông báo lỗi nếu KBUILD\_OUTPUT rỗng.

**Chỉ Định CROSS\_COMPILE.**

# set default to nothing for native builds

ifeq ($(HOSTARCH),$(ARCH))

CROSS\_COMPILE ?=

*endif*

Đoạn mã trên xác định xem mã nguồn uboot sẽ được biên dịch native hay cross compile. Mặc định là native builds.

**Phân Tích make xxx\_defconfig.**

1. %config: scripts\_basic outputmakefile FORCE
2. $(Q)$(MAKE) $(build)=scripts/kconfig $@

Tương tự như menuconfig. %config chính là điểm vào của make xxx\_defconfig.

Đối tượng chúng ta đề cập ở đây là board BBB với lệnh cụ thể là :

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf am335x\_boneblack\_vboot\_defconfig

Chúng ta có thể mở rộng luôn câu lệnh ở đây.

am335x\_boneblack\_vboot\_defconfig: scripts\_basic outputmakefile FORCE

$(Q)make –f $(srctree)/scripts/Makefile.build obj=scripts/kconfig   
 am335x\_boneblack\_vboot\_defconfig

Mở ./scripts/kconfig/Makefile và chúng ta sẽ thấy

%\_defconfig: $(obj)/conf

$(Q)$< $(silent) --defconfig=arch/$(SRCARCH)/configs/$@ $(Kconfig**)**

Mục tiêu này phù hợp với am335x\_boneblack\_vboot\_defconfig tuy nhiên nó khác vơi menuconfig. Dependence của nó không còn mà mconf mà là conf. Conf cũng giống như mconf đều là chương trình C được biên dịch và chạy trên máy này. Chúng ta mở rộng lệnh như sau :

$(Q)/scripts/kconfig/conf --defconfig=arch/../configs/am335x\_boneblack\_vboot\_defconfig Kconfig.

Chương trinh conf sử dụng hai tham số là –defconfig và Kconfig. Chương trình này không có giao diện đồ họa, sẽ hoạt động dựa trên tệp cấu hình chỉ định và cuối cùng tạo ra .config.